Amendment to the Specification:

Please replace paragraph 0008 on page 3 with the following rewritten paragraph:

[0008] In principal, both the old and new standards, however, rely on cyclic redundancy check (CRC) ~~to~~ for determining when errors are present in a data stream. A CRC performs a mathematical calculation on a data stream before and after data transmission. If the two results are identical, then it is assumed that no errors occurred during transmission.

Please replace paragraph 0013 beginning on page 5 with the following rewritten paragraph:

[0013] The CRC calculation ends when there is no more data to input into the circuit. The final values in the delay flip flops are collectively called the residue, which is equal to the remainder of the data stream divided by the generator polynomial 1010000000000011 in mod-2 arithmetic. It should be noted that the values of C0, C1, C2 ... C15 are zero in this circuit 200. As will be seen, C0, C1, C2 ... C15 are only relevant for ~~byte~~bit-wise calculations.

Please replace paragraph 0020 on page 8 with the following rewritten paragraph:

[0020] Like any remainder, the CRC result represents how far away numerically the dividend is from being evenly divisible by the divisor. In regular division, the remainder must be subtracted from the dividend in order to be evenly ~~divisibly~~ divisible. For example, 17 divided by 4 gives a remainder of 1, which must be subtracted from 17 in order to make 16 and be evenly divisible by 4. (There are, of course, other relationships between the dividend, remainder and divisor, which are not important for this discussion.) However, in modular division, the remainder can be added to the dividend in order to make the number evenly divisible by the divisor. CRC transmitting circuits use this relationship by adding the remainder to the dividend so that the CRC receiving circuit will calculate a zero remainder if no error is present.

Please replace paragraph 0026 on page 10 with the following rewritten paragraph:

[0026] The present invention provides a system and method of performing a CRC calculation on multiple bytes of data in a single cycle. The system includes a multiple-byte CRC circuit, which in one embodiment, may comprise a first CRC module, a second CRC module and a decision module.

Please replace paragraph 0027 on page 10 with the following rewritten paragraph:

[0027] The first CRC module performs the CRC operation on a maximum number of bytes of data in a cycle and produces a result. The maximum number of bytes will often be eight, but there may be applications where it is desirable to process more or less bytes of data in each cycle. The second CRC module performs the CRC operation on a single byte of data, and also produces a result. Both CRC modules are capable of using prior results when performing their CRC operation. Being able to use the result of a prior cycle ensures that a long string of data can be accurately processed in multiple cycles. The decision module directs data to the ~~first~~ second CRC module only when the number of remaining bytes of data to be processed is greater than or equal to the maximum number of bytes.

Please replace paragraph 0028 on page 10 with the following rewritten paragraph:

[0028] ~~A~~ In a specific embodiment of the invention, the multiple-byte CRC circuit uses a total of eight CRC modules. Each CRC module processes a different number of bytes, with eight being the maximum number of bytes. In this specific embodiment, the decision module directs the data to one of the eight CRC modules, depending on the number of bytes of data to be processed.

Please replace paragraph 0029 beginning on page 10 with the following rewritten paragraph:

[0029] The first step in the method of performing a CRC calculation on data is providing ~~multiple~~ a plurality of CRC ~~circuits~~modules, at least one CRC ~~circuit~~ module being able to perform the CRC calculation on a maximum number of bytes of data and at least one CRC ~~circuit~~ module able to perform the CRC calculation on only a single byte of data. The next step is determining which one of the ~~multiple~~ plurality of CRC ~~circuits~~ modules is appropriate for processing a number of bytes. The next step is processing the number of bytes with the appropriate CRC ~~circuit~~ module. Finally, the steps of

determining which CRC ~~circuit~~ module is appropriate and processing the number of bytes with the appropriate CRC ~~circuit~~ module are repeated until there are no more bytes to process.

Please replace paragraph 0046 on page 15 with the following rewritten paragraph:

[0046] Transmit 66/64b converter circuit 620 converts a 66-bit data stream into a 64-bit data stream. The relationship between the clock rate of the signal entering transmit 66/64b converter circuit 620 and the clock rate of the signal leaving circuit 620 is the exiting signal is 33:32. No data is lost as long as 33 cycles of 64-bit data takes the same amount of time to propagate as 32 cycles of 66-bit data. Similarly, receive 66/64b converter circuit ~~350~~ 650 slows down and expands a 64-bit data stream into a 66-bit data stream.

Please replace paragraph 0049 beginning on page 15 with the following rewritten paragraph:

[0049] FIG. 7A shows an exemplary method for using a multiple byte ~~wise~~ CRC circuit ~~that can~~, which includes a plurality of CRC modules that are configured for performing CRC calculations on up to eight bytes at a time. In one embodiment, the plurality of CRC modules may include a total of eight CRC modules, each configured for performing a CRC calculation on a different number of bytes of data at a time (i.e., during a single cycle). For example, an eight-byte-wide CRC module may be included for performing a CRC calculation on an eight-byte-wide segment of data, a seven-byte-wide CRC module may be included for performing a CRC calculation on a seven-byte-wide segment of data, etc. First, in step 710 of the method, the storage is reset to zero in order to ensure that CRC values from a previous run do not influence the calculations.

Please replace paragraph 0050 on page 16 with the following rewritten paragraph:

[0050] Next, in step 720, a determination is made as to whether there are more than eight bytes remaining in the data stream. If there are more than eight bytes remaining, the process proceeds to step 730, where an eight-byte CRC calculation is performed on the data using, e.g., an eight-byte-wide CRC module.

Please replace paragraph 0051 on page 16 with the following rewritten paragraph:

[0051] An eight-byte-wide CRC ~~circuit~~ module can be constructed ~~by~~ using the same general process as was described by Aram Perez for byte-wise CRC circuits. Namely, a bit-wise CRC circuit is first modeled, then the general values of each flip flop after sixty-four bits are calculated and, finally, a ~~circuit~~ module that implements the calculations is created. The resultant eight-byte-wide CRC module would be configured to perform an eight-byte CRC calculation on eight bytes of data and to produce results of such calculation, all in a single cycle. Alternatively, once a byte-wise CRC ~~circuit was~~ module is modeled for the appropriate generator polynomial, it ~~would~~ could be repeatedly used to generate the same general values after 8 bytes (i.e., after eight cycles), as a bit-wise CRC circuit would generate after 64 bits (i.e., after 64 cycles). However, since a byte-wise CRC module is able to process only one byte at a time, it would take seven more cycles than the eight-byte-wide CRC module to produce the same result. Appendix I shows the results from such a calculation after using a CRC-32 generator polynomial.

Please replace paragraph 0052 beginning on page 16 with the following rewritten paragraph:

[0052] In step 740, the results of the eight-byte CRC calculation are stored in memory. Although initially zero, the storage will likely change after every cycle. It should be noted that a very similar process could be used in a single byte-wise CRC circuit or module ~~a very similar process would be used~~. Namely, a byte-wise CRC circuit would initialize the system in a step similar to step 710, and steps similar to steps 730 and 740 would be repeated until no more data was present. Of course, the byte-wise corollary to step 730 would perform a byte-wise CRC calculation, ~~and~~ not an eight-byte CRC calculation, and thus, would need seven more cycles to produce the same results. Therefore, one having ordinary skill in the art should appreciate that techniques currently known in the art for byte-wise CRC calculations can be easily implemented in connection with steps 710, 730 and 740.

Please replace paragraph 0054 on page 17 with the following rewritten paragraph:

[0054] In step 750, one of eight CRC ~~circuits~~ modules is used to calculate the CRC value on the remaining bytes of data. If there is only one byte remaining, a single-byte CRC ~~circuit~~ module would need to be used. If there are two bytes remaining, a two-byte CRC ~~circuit~~ module would be used, etc. Each CRC ~~circuit~~ module could be constructed ~~by~~ using the same general process of modeling known CRC circuits, and then generating general values for the appropriate number of bytes to be processed per

cycle. After an initial processing step is performed, Tthe inputs to the appropriate ~~circuit~~ CRC module (i.e., the CRC module configured for processing the number of remaining bytes of data) would ~~be~~ include the remaining bytes of data and the results from the prior CRC calculations, which were stored in step 730. Appendix II shows the general results of two bytes after modeling a CRC ~~circuit~~ module using a CRC-32 generator polynomial. Appendix III shows the general results of three bytes after modeling a CRC ~~circuit~~ module using a CRC-32 generator polynomial. Appendix IV shows the results of four bytes, Appendix V shows the results of five bytes, Appendix VI shows the results ~~from~~ of six bytes and Appendix VII shows the results ~~for~~ of seven bytes.

Please replace paragraph 0055 beginning on page 17 with the following rewritten paragraph:

[0055] FIG. 7B shows an alternate method for performing a CRC calculation on data, according to another embodiment of the invention. In FIG. 7B, Ssteps 710 through 740 are identical to steps 710 through 740 of FIG. 7A. However, Hinstead of sending the remaining bytes of data to one of the eight different ~~circuits~~ CRC modules, as was done in step 750 of FIG. 7A, only one single-byte CRC ~~circuit~~ module is used to process the remaining bytes of data (in step 760). After the single-byte CRC module performs a CRC calculation on the first byte of the remaining bytes of data, Tthe process ~~then~~ progresses to step 770, where a determination is made as to whether there are any more bytes to process. Therefore, step 760 ~~might~~ may potentially be repeated up to eight times, which is seven cycles more than the process described in FIG. 7A. In certain applications, the extra time required to calculate the CRC of the data might be preferable to the cost of the extra circuitry. The maximum number of times step 760 repeats itself could be reduced ~~to seven~~ if the same eight-byte CRC ~~circuit~~module, ~~that is~~ which was used in step 730, is used ~~if~~ when there are exactly eight bytes of data remaining. Using the same eight-byte CRC circuit module would also be preferable in the method described in FIG. 7A in order to reduce the cost of the circuit.

Please replace paragraph 0056 on page 18 with the following rewritten paragraph:

[0056] A possible compromise between a large circuit and slow processing times might be a system that uses an eight-byte, a three-byte and a ~~single byte~~ single-byte CRC ~~circuit~~ module. Such a method would use significantly less chip area than the method described in FIG. 7A, and would ~~only~~ take a maximum of two extra cycles to complete the CRC calculation.